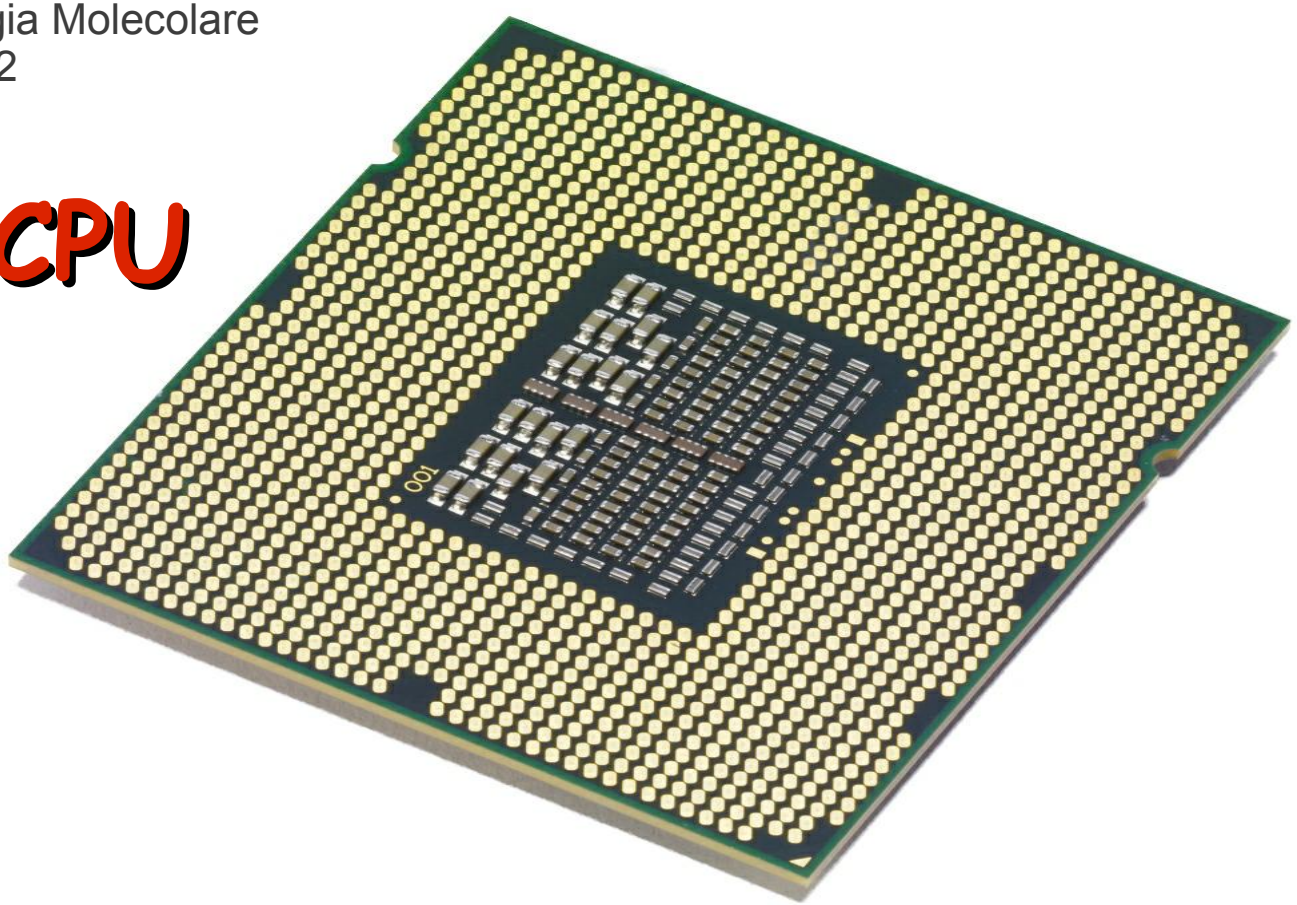
Corso di Laurea in Biologia / Biologia Molecolare
Università di Padova, AA 2011/2012

Laboratorio CPU



Docente: Ivilin Stoianov
Assistenti: Michele De Filippo, Alberto Testolin

Lab P150 (Paolotti)
sito web: www.stoianov.it
Email: info@stoianov.it

Commandi relativi al percorso

pwd indica la posizione attuale nel file system

ls visualizza il contenuto di una cartella o i file

ls [opzione] [nome_directory] [nome_file]

-a elenca tutti i file compresi quelli nascosti

-l fornisce informazioni aggiuntive

cd cambiare la cartella (la posizione nel file system)

cd [nome-directory]

cd .. sale di un livello nell'albero

cd ritorna all propria home

mkdir creare una cartella

mkdir [opzioni] nome_directory

rmdir eliminare una cartella

rmdir nome_directory

info@stoianov.it

Creare e visualizzare file di testo

pico semplice editore di testo (o "**nano**")

pico [nome_file]

Nell'editore: **Ctr-O** salva il file; **Ctr-X** esce dal editore

more visualizza il contenuto di un file

more nome_file

Tasti: **barra-*spazio*** per successiva pagina; "**Q**" per uscire.

less visualizza il contenuto di un file

Tasti: **frecce** per navigare; "**Q**" per uscire.

less nome_file

ESERCIZIO

- spostarsi nella cartella "Documents"
- utilizzando l'editore ***pico***, creare un file di testo "prova.txt" e scrivere dentro qualche righa di testo
- salvare il file ed uscire dall'editore
- visualizzare il contenuto del file "prova.txt" utilizzando "more" e "less"

info@stoianov.it

Dupplicare ed eliminare dei file

cp copiare una cartella o file

- in una cartella

cp [opzioni] file_origine nome_directory

- con un nuovo nome

cp [opzioni] file_origine file_destinazione

Esempio: cp mydoc.txt cartella

rm eliminare una o più cartelle o file

rm [opzioni] nome-file

man manuale dei commandi di linux (o altre applicazioni)

Navigare con le **frecce** ed uscire con tasto "**Q**"

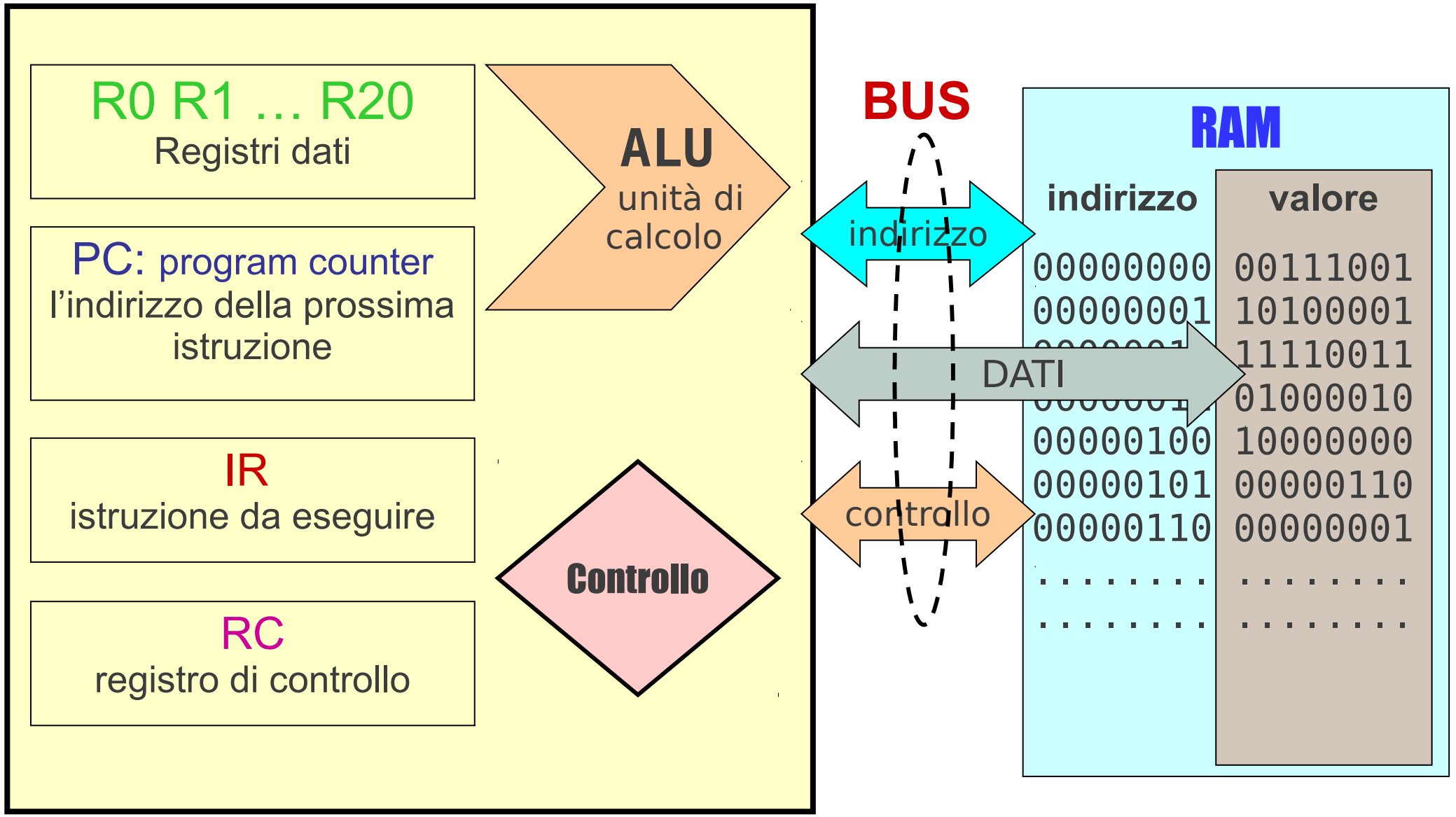
man nome-commando

ESERCIZIO

- dopo aver creato un file di testo "prova.txt" (nella cartella "*Documents*"), duplicare il file con il nuovo nome 'prova2.txt' nella stessa cartella
- copiare il file prova2.txt in nella cartella "biologia", che è una sottocartella della cartella "Documents"
- eliminare il file "prova.txt"

info@stoianov.it

CPU



info@stoianov.it

microcalc

simulatore di un CPU e interpretatore di un relativo linguaggio di programmazione al livello CPU (Assembler)

- **istruzioni mnemonici** (**LOAD, STORE, ADD**, …)
- **memoria** accessibile tramite **variabili**: **X: INT 10**
- **istruzioni** indirizzabili con **eticetta**: Good: **FINE**
- Il programma va scritto in un semplice file testuale:

```
nano un-programma.cpu
```

- un'istruzione per riga, ogni riga finisce con **;**
- dopo "**;**" potrebbe seguere un **commento**
- ogni programma contiene:
  - **blocco definizione dati**
  - **una riga vuota**
  - **blocco istruzioni**

info@stoianov.it

microcalc (2)

- Prima di eseguire un programma, *microcalc* visualizza la sua interpretazione del programma e dello stato dei registri e delle variabili dechiarati all'inizio del programma.
- Nel caso di errore nel programma, *micrcalc* lo segnala e ferma l'esecuzione del programma.
- Se non rileva errori, *microcalc* esegue il programma e alla fine stampa l'ultimo stato delle variabili e dei registri.
- Se viene eseguito in modalità “**debug**” (con “-d”), *microcalc* visualizza lo stato delle variabili e dei registri dopo l'esecuzione di ogni istruzione.

microcalc (3)

Per accedere al simulatore:

1) creare una cartella prg (per scrivere dentro programmi) ed entrare dentro.
2) verificare se *microcalc* funziona, scrivendo sul terminale:

   microcalc

3) si potrebbe quindi scrivere un programma "*programma.cpu*" con un semplice editore di testo (per es., *nano*) ed eseguirlo:

   microcalc *programma.cpu*

microcalc [-d] <nome programma>
v.2.1-20110530 v.1.0: A.Ceccato; v.2.0+: I.Stoianov

Simulatore di un CPU. Registri R0...R19. Variabili int e float.

...

Il primo programma (**trasferimento dati)**

oBBIETIVO:

- imparare a scrivere un programma;
- imparare ad ***eseguire il programma*** con il simulatore **microcalc**;

COMPITO: Assegnare la costante 10 ad una variabile *X*

- scrivere un programma in un file testuale "**1-load.cpu**"
- eseguire il programma tramite l'interpretatore **microcalc**:

  microcalc 1-load.cpu
- eseguire il programma passo per passo

  microcalc -d 1-load.cpu
- osservare come cambiano i registri e la memoria ad ogni passo

---

```
X: INT;        definire una variabile X

LOAD   R0 10; caricare la costante 10 nel Registro R0
STORE  R0 X;  salvare il valore del R0 nella variabile X
STOP;         fine programma
```

Primi passi: **Aritmetica**

OBBIETIVI:

- imparare a dichiarare variabili, assegnandone subito un valore
- imparare ad usare registri per fare calcoli

Compito: X=X+Costante (X=X+1) **2-add**

- definire una variabile ed assegnarne un valore iniziale (X=10)
- caricare la variabile **X** in un registro **R0**
- aggiungere al registro la costante **1**
- assegnare il risultato (nel registro) alla variabile **X**

```
X: INT 10;     Definire una variabile tipo int; assegnarne valore 10

LOAD R0 X;     R0=X
ADD R0 1;      R0=R0+1
STORE R0 X;    X=R0
STOP;          FINE del Programma
```

Primi passi: **Aritmetica con 2 variabili**

OBBIETIVO:

- imparare ad eseguire calcoli con due registri dati

Compito: X=X+Y **3-add**

- definire due variabili, X e Y, e darne valori iniziali
- caricare X in registro R0 e Y in registro R1
- aggiungere R1 ad R0
- copiare R1 in X

```
X: INT 10;    X=10
Y: INT  2;    Y=2

LOAD R0 X;    R0=X
LOAD R1 Y;    R1=Y
ADD R0 R1;    R0=R0+R1
STORE R0 X;   X=R0
STOP;         FINE del Programma
```

Diagrammi di Flusso

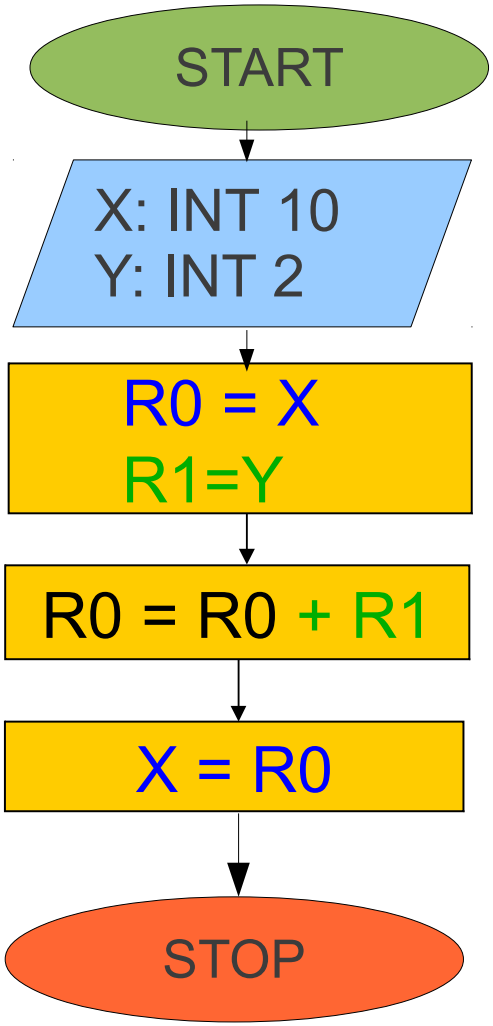
- Per capire la logica dei programmi con *cambio del flusso* è utile rappresentarla in modo visuale.

- La rappresentazione grafica più comune: **Diagrammi di Flusso**.
- Costruiti dai seguenti blocchi:

- Ogni passo elementare del algoritmo è rappresento con un blocco.
- Un blocco "**A**" potrebbe essere collegato tramite freccia con un altro blocco "**B**", indicando che "**B**" dovrebbe essere eseguito dopo "**A**".
- Un blocco decisionale è collegato con due o più blocchi.
- La sequenza dei passi che costruiscono l'algoritmo si codifica con la sequenza dei blocchi, collegati con frecce.

Flusso lineare (1)

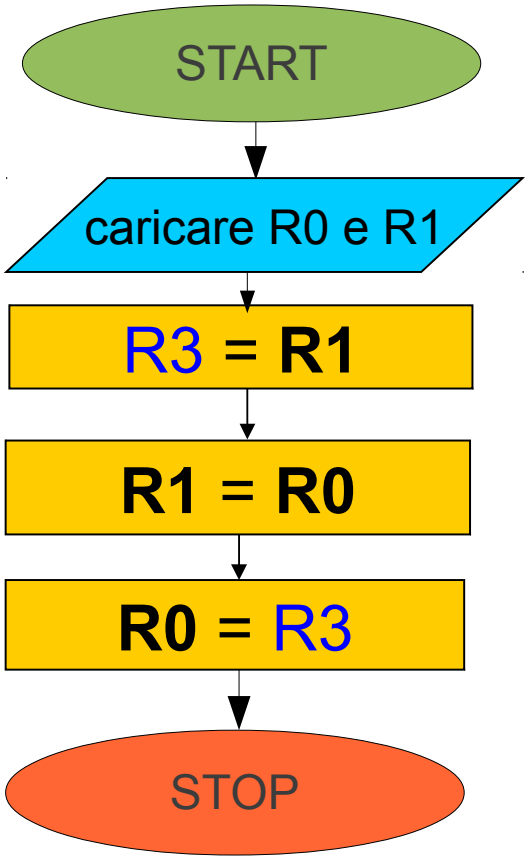
Esempio: Aggiungere Y al X

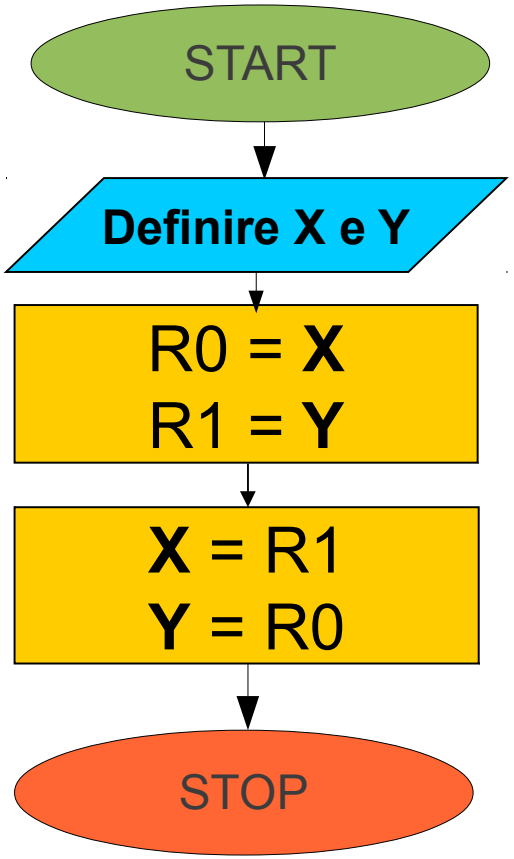



```
                    ; DATI
X:INT 10            ; X=10
Y:INT  2            ; Y=2
                    ; PROGRAMMA
LOAD   R0,X         ; X→R0
LOAD   R1,Y         ; Y→R1
ADD    R0,R1        ; R0=R0+R1
STORE  R0,X         ; R0→X
STOP                ; fermi !
```

Flusso lineare (2)

ESERCIZIO
Scambiare il contenuto di:
(a) due registri R0 e R1
(b) due variabili X e Y





COMPITO

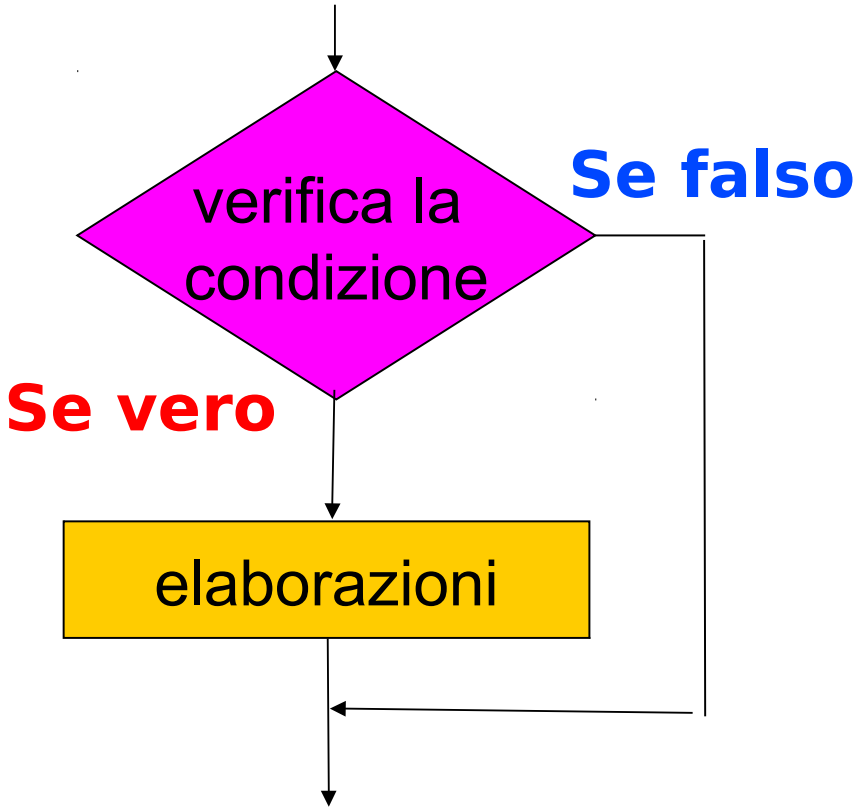
Scrivere due programmi
in Assembler
per il nostro CPU
che realizzano gli algoritmi

info@stoianov.it

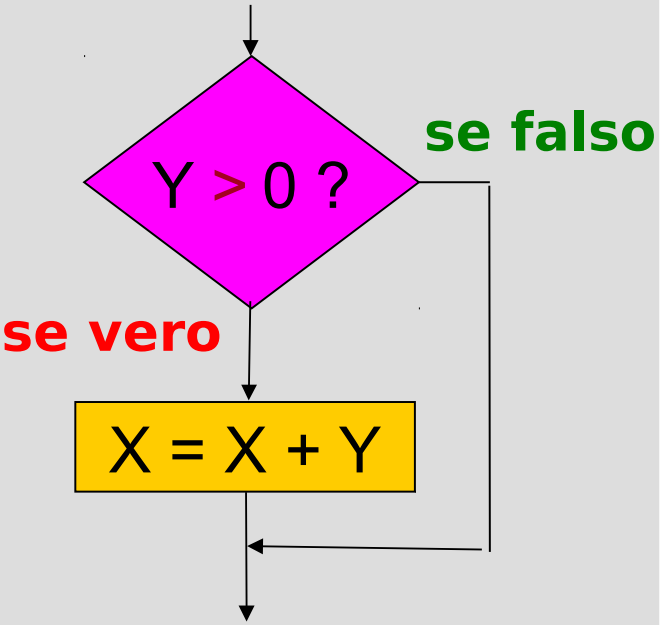
4-scambio-registri
5-scambio-variabili

Flusso non-lineare: Elaborazione condizionale ad una via

Eseguire una serie di elaborazioni solo se una condizione è vera



Esempio:
Aggiungere Y ad X
solo se Y>0



Confronto

- COMP ***r1***, ***r2*** confronta i valori-***integer*** dei registri ***r1*** e ***r2***

  Risultato: Se Rr1<Rr2 , memorizza -1 nel registro ***RC***
  Se Rr1=Rr2 , memorizza 0
  Se Rr1>Rr2 , memorizza +1

- COMP ***r1***, ***cost*** confronta il registro ***r1*** con *la costante* ***cost***

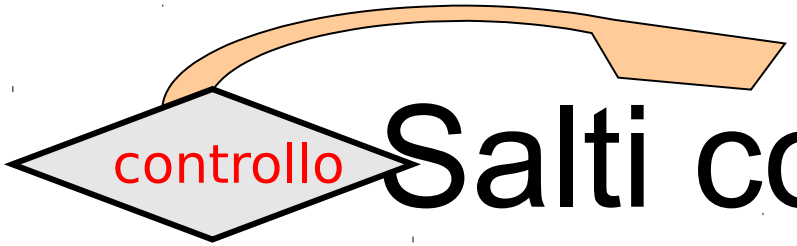
- FCOMP ***r1***, ***r2*** confronta valori del tipo ***float***

COMP Operazione registro1 registro2 bit non-utilizzati

```
[0010 0000][iiii][jjjj] [xxxxxxxx xxxxxxxx]
```

COMP Operazione registro costante

```
[1010 0000][iiii][xxxx xxxx xxxx xxxx xxxx]
```



Salti condizionali

- BRLT ***addr*** se RC=-1, spostare il PC all'***addr***
- BRLE ***addr*** se RC<=0, spostare il PC all'***addr***
- BREQ ***addr*** se RC = 0, spostare il PC all'***addr***
- BRNE ***addr*** se RC ≠ 0, spostare il PC all'***addr***
- BRGT ***addr*** se RC = 1, spostare il PC all'***addr***
- BRGE ***addr*** se RC>=0, spostare il PC all'***addr***

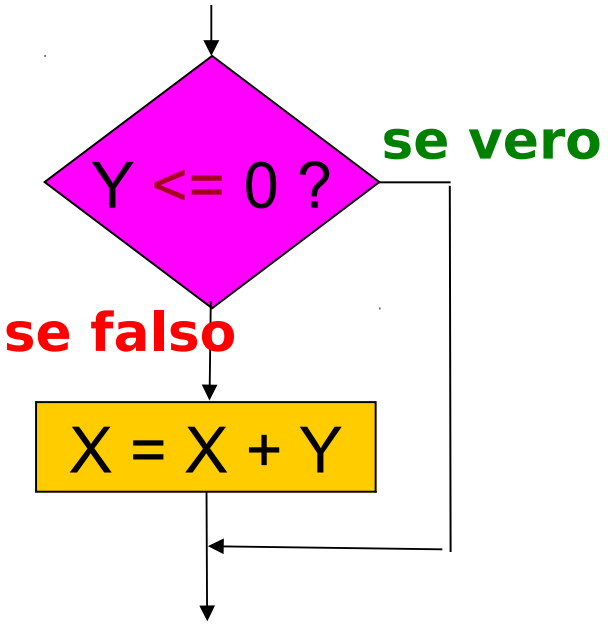
BRLT Operazione bit non-utilizzati nuovo indirizzo per il PC

```
[0100 0001][xxxx][aaaa aaaa aaaa aaaa aaaa]
```

Elaborazione condizionale ad una via utilizzando il complimento della condizione

Per realizzare l'esecuzione condizionale (per es., "**A**" è vera) di un **blocco di elaborazioni**, semplicemente saltare il **blocco di elaborazioni** se la condizione complimentare (**not A**) è vera.

Esempio:
Aggiungere Y al X
solo se Y>0




```
X: INT 5;
Y: INT -3;

LOAD R0 X;
LOAD R1 Y;
COMP R1 0;      Y<=0 ?
BRLE Fine;      YES? → FINE
ADD R0 R1;      Acc=Acc+X
STORE R0 X;     Y+X → X
FINE: STOP;
```


info@stoianov.it